

Using Wax and Jekyll to build minimal digital projects

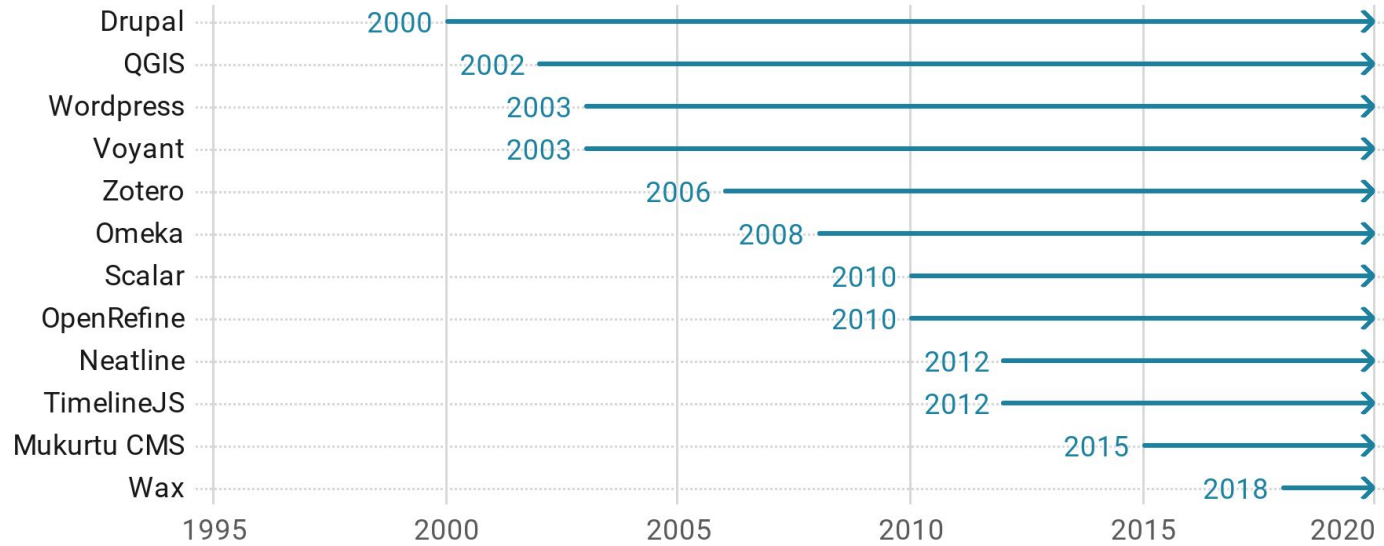
Jo Klein

Cass Wilkinson Saldaña

Angela Zoss (presenting)



Open source digital scholarship tools over time



Created with Datawrapper



Why build digital projects minimally?

Rising interest in digital scholarship over recent decades

Challenges:

- Maintenance and ongoing labor
- Learning curve
- Access
- Accessibility
- Autonomy (hosted vs locally managed instances, open source)
- Surveillance
- Justice



“I prefer to approach minimal computing around the question **“What do we need?”**”

...we aim to understand ways of building that could be referred to as **architectures of necessity...**”

[“The User, the Learner and the Machines We Make”](#)
by Alex Gil (2015) via Minicomp



What is Minicomp?

“Minicomp is a collaborative effort in embedding minimal computing principles (e.g. [Minimal Maintenance](#), [Minimal Dependencies](#), and [Minimal Connectivity](#)) in digital humanities methods.”

[Minicomp Wiki](#)



Minimal computing values in digital projects

Minimal:

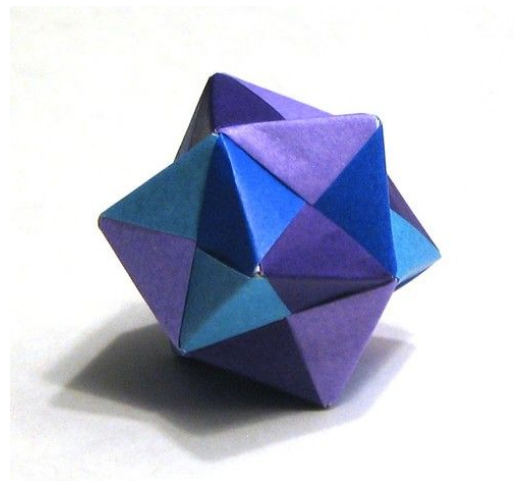
- Maintenance
- Dependencies
- Obsolescence

Maximum:

- Access
- Accessibility
- Ephemerality

[Minimal Definitions \(tl;dr version\),](#)

Jentery Sayers - 03 Oct 2016

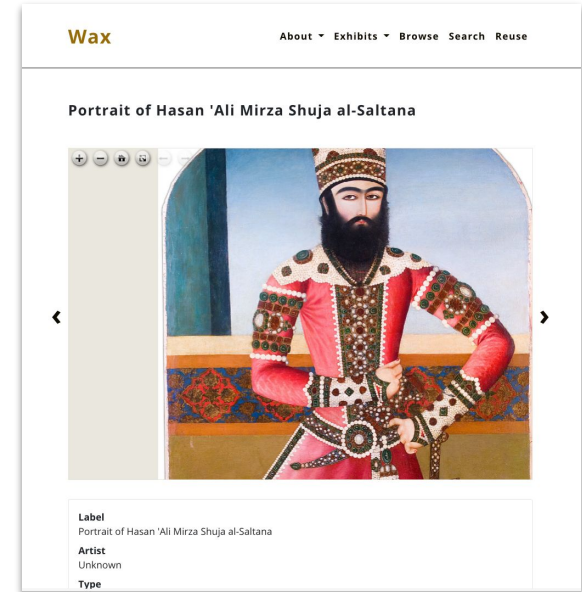
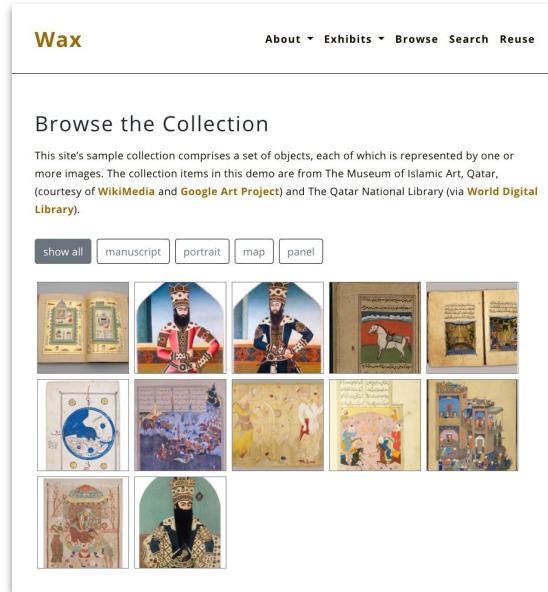


["Origami stellated octahedron"](#) by [endolith](#)
is licensed under [CC BY-SA 2.0](#)



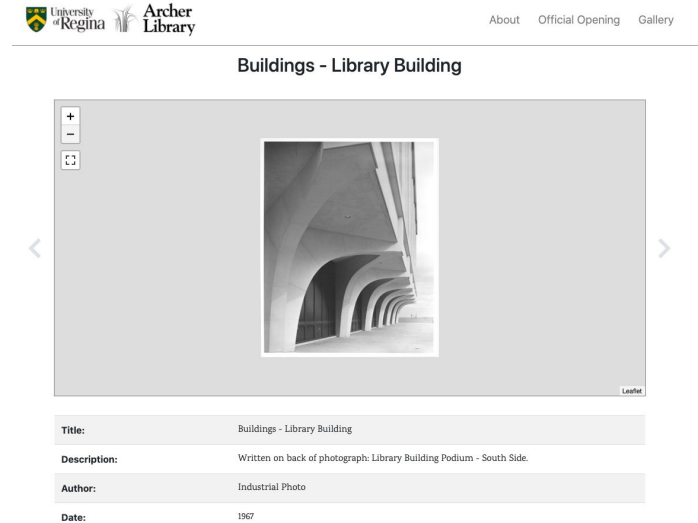
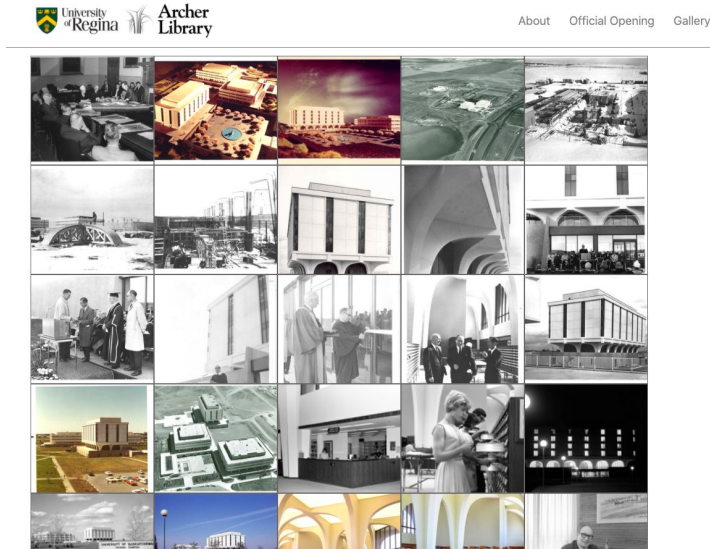
What is Wax?

Wax builds on an existing website platform to create minimal digital exhibits.



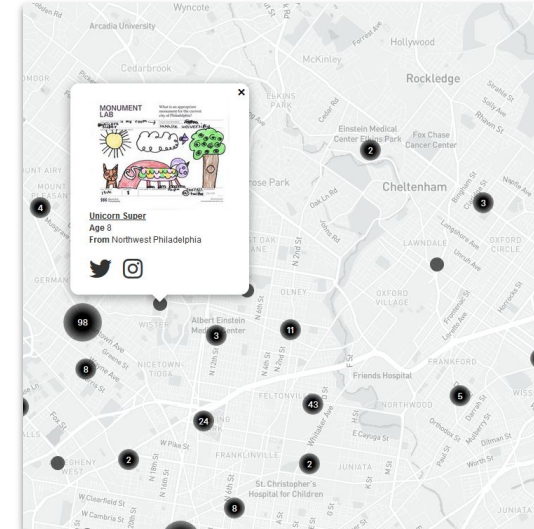
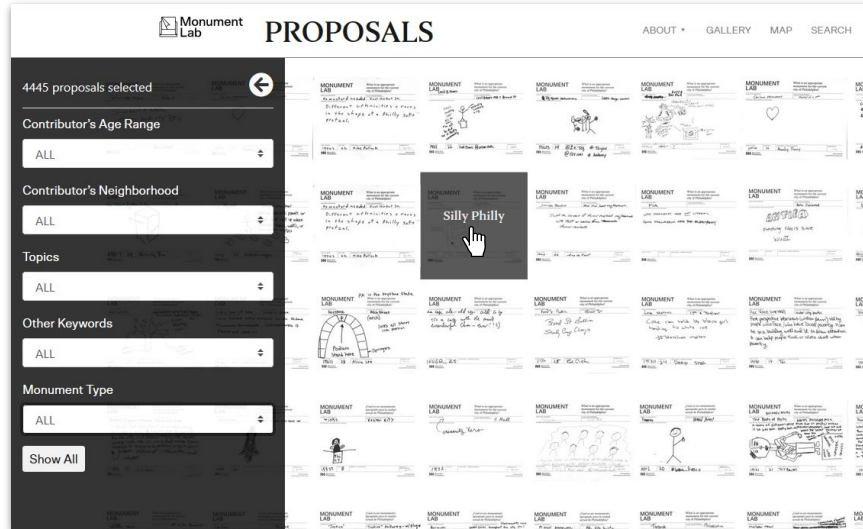
Archer Library - 50th Anniversary Town Hall

University of Regina - Archer Library



Proposals: Creative Speculations For Philadelphia

Monument Lab & Mural Arts Philadelphia



How Does Wax Work?



First, a note about Jekyll

- Compiles plain text files into a full website
- Built on Ruby
- Has built-in concepts of pages, posts, and (important for Wax) **collections**



Wax saw Jekyll as a good foundation for digital exhibits.

- Storing objects
- Displaying objects
- Keeping it minimal

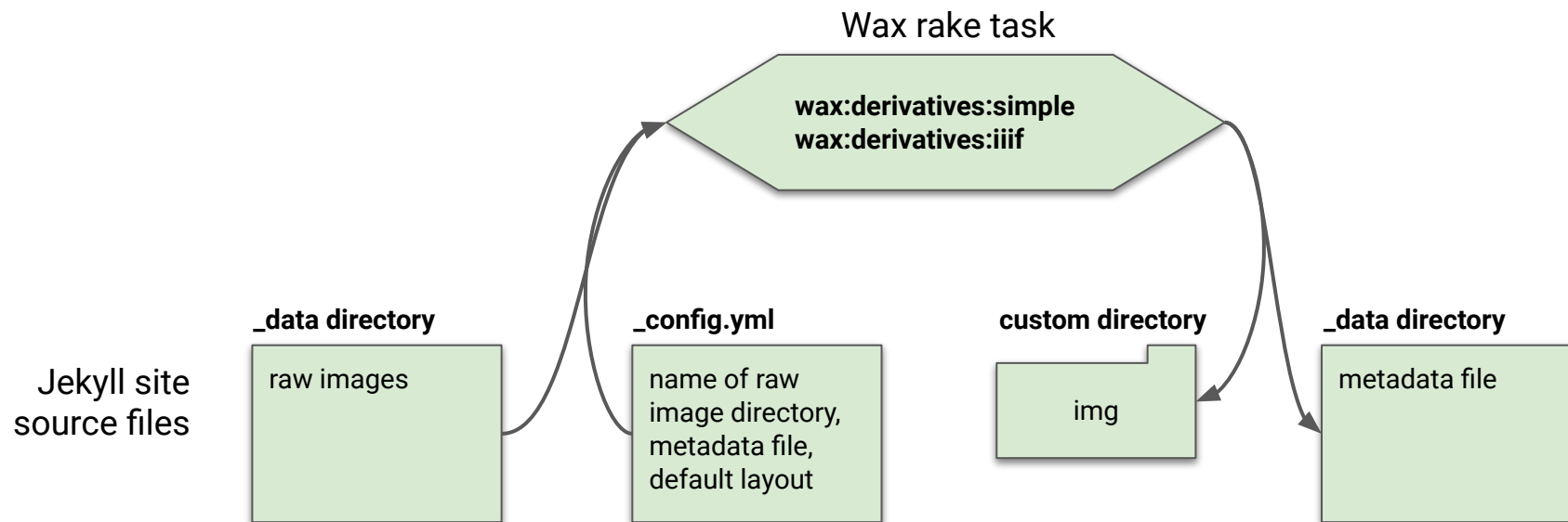


Wax helps prepare digital exhibits for Jekyll.

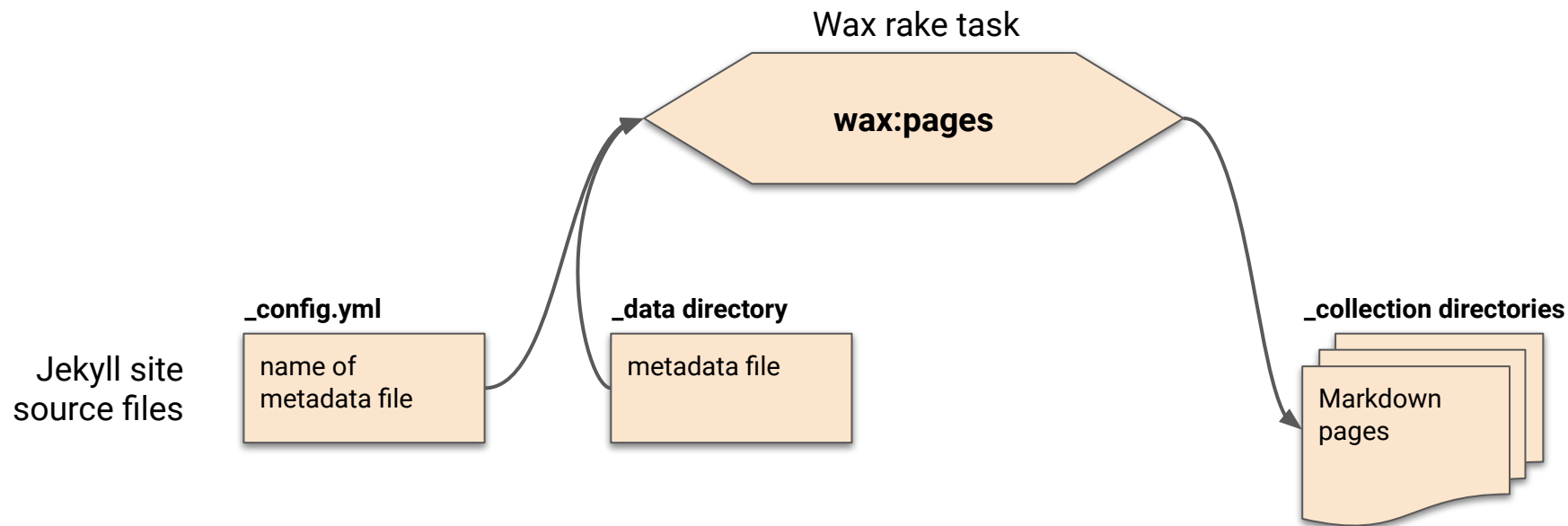
- Processing images
- Generating item pages
- Adding search functionality



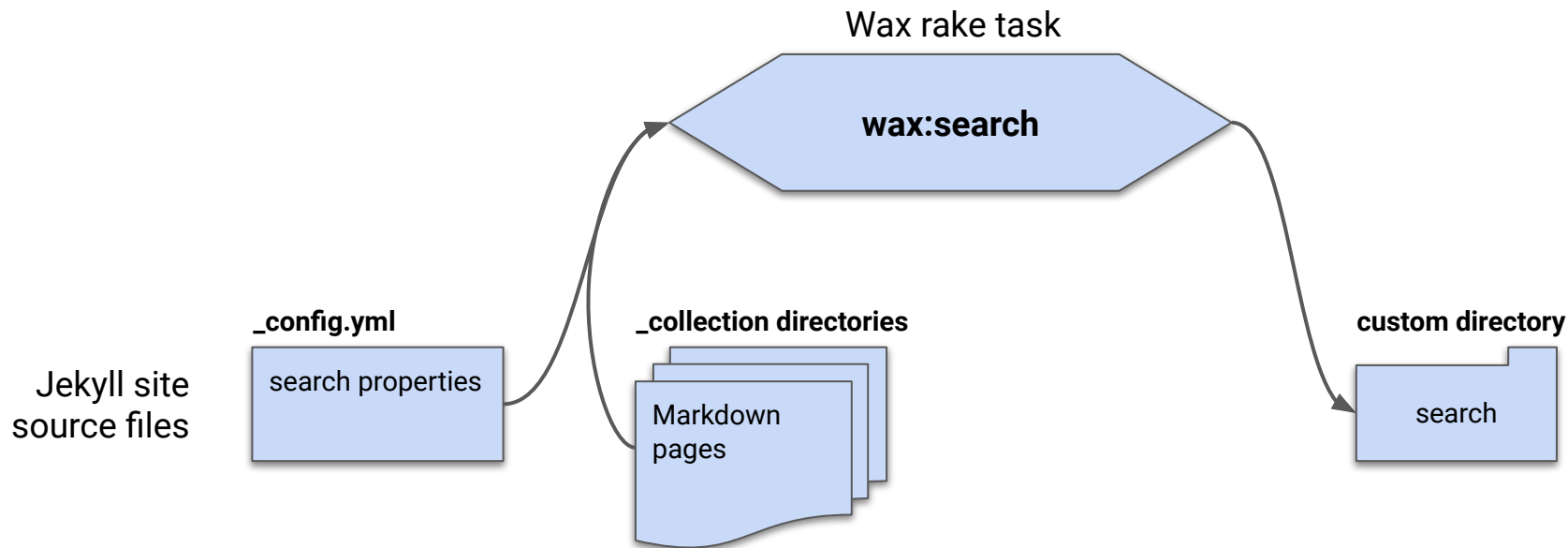
Wax image processing



Wax page generation



Wax search functionality



Special pages that come with Wax



“Browse” pages/include

- Repository has a page called “collection”
- Collection page uses an include called “collection_gallery”:
 - pulls in collection metadata
 - builds an image gallery
 - adds filter buttons at the top

Wax

About ▾ Exhibits ▾ Browse Search Reuse

Browse the Collection

This site's sample collection comprises a set of objects, each of which is represented by one or more images. The collection items in this demo are from The Museum of Islamic Art, Qatar, (courtesy of [WikiMedia](#) and [Google Art Project](#)) and The Qatar National Library (via [World Digital Library](#)).

show all manuscript portrait map panel



“Item” pages/layout

- Wax task creates item pages, including default layout from configuration file
- Repository has layout called “generic_collection_item”
 - image viewer at the top
 - table of metadata at the bottom

Wax

About ▾ Exhibits ▾ Browse Search Reuse

Portrait of Sheikh Ali Mirza



Label
Portrait of Sheikh Ali Mirza
A. Mirza



Easiest Wax Project

- Download files from the Wax demo repository
- Create a .csv and add to _data folder
- Create raw images and add to _data/collection folder
- Edit configuration file to add collection info, search info, other site information
- Run Wax tasks
- Edit Jekyll pages to update content for current project
- Create a GitHub repository using files
- Set up GitHub Pages for hosting

Skills that end up being useful

- Installing software dependencies
- Running and troubleshooting command line tasks
- Creating and maintaining a GitHub repository



Sample Project: Teaching Viz By Example

*This project was made possible in part by
the [Institute of Museum and Library Services](#),
RE-73-18-0059-18.*



Teach Viz by ExampleUser Guide Search Browse ▾

Welcome to the Teach Viz by Example repository!

This is a space for the data visualization teaching community to find, share, and contribute exemplary data visualizations + visualization-ready datasets.

Please explore our beta release and let us know what you think! Look for an option to contribute new material soon.

Select an option below to get started

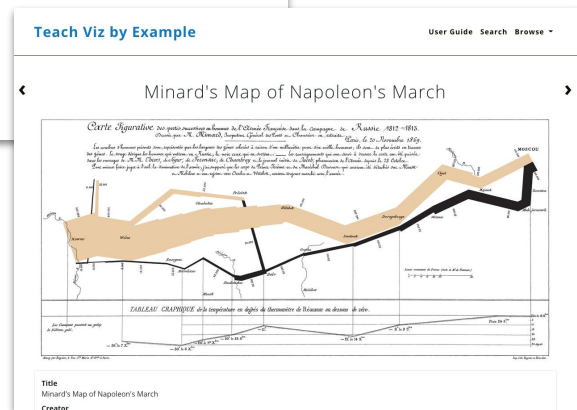
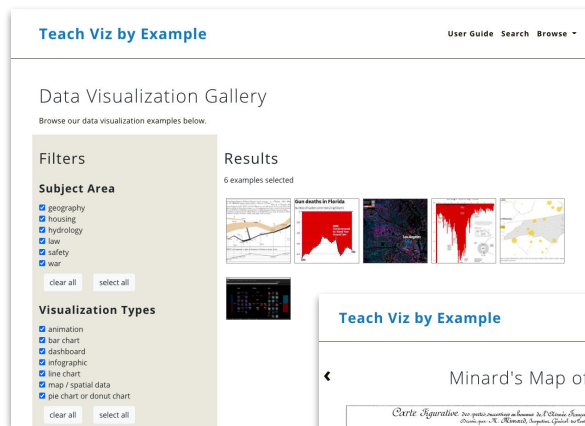
[Browse data visualization examples](#)[Browse dataset examples](#)[Review our how-to guide](#)

[Teach Viz by Example](#)



Why Wax?

- Minimal computing philosophy
- Lightweight, easy to update and maintain
- Host for free on GitHub
- Open to some programming



How we are extending Wax

Lists in fields

Visualization Types

- line chart
- map / spatial data

Multi-facet filters

Visualization Types

- ☒ animation
- ☒ bar chart
- ☒ dashboard
- ☒ infographic
- ☒ line chart
- ☒ map / spatial data
- ☒ pie chart or donut chart

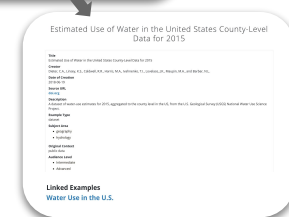
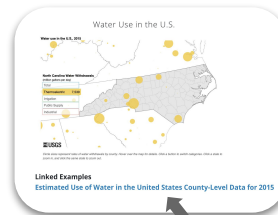
clear all

select all

Data dictionary

```
field_label: 'Date of Creation'
- field_name: source_url
field_label: 'Source URL'
type: link
- field_name: description
field_label: 'Description'
- field_name: example_type
field_label: 'Example Type'
- field_name: subject_area
field_label: 'Subject Area'
- field_name: original_context
field_label: 'Original Context'
- field_name: audience_level
field_label: 'Audience Level'
- field_name: audience_composition
field_label: 'Audience Composition'
- field_name: pedagogical_description
field_label: 'Pedagogical Description'
- field_name: ethical_quandaries
```

Linked items



Pro-Tips

- Cloning is different from downloading! Stick with downloading.
- Wax derivatives tasks update the metadata file, so keep a backup
- Wax doesn't update metadata if image derivatives already exist, so delete those before rerunning derivatives
- Wax doesn't regenerate pages, so delete those before rerunning pages task



Resources for learning more

Our grant and project work

- [Visualizing the Future](#)
- [Examples Repository GitHub](#)
 - Sample [Google Apps Script](#) for list processing
 - [Python processing script](#)

More on Jekyll

- [Why Use a Static Site Generator?](#)
- [Main Jekyll Documentation](#)
- [Liquid Template Language](#)

Writing and Maintaining Code

- Free code editors: [Atom](#), [Visual Studio Code](#)
- [GitHub Desktop](#) to maintain GitHub repository
- [Creating a GitHub Pages site with Jekyll](#)

More on Wax

- [Wax wiki](#) = ample resources to create your own instance
- [Minimal computing for image collections: the case of wax](#) by alex gil, marii nyröp @ DLF2018 Minimal Computing Panel

Advanced but useful

- [Bootstrap](#) for website layout
- [jQuery](#) for website functionality
- [Ruby](#), if you want to go really deep



Thank you!

Email us at visualizingthefuture@umich.edu



How we are extending Wax: support for list fields

- Easier to create metadata in spreadsheet, but have some fields that are natural lists (e.g., “visualization type”, “audience level”)
- Created python script to take .csv, split lists into arrays, and output JSON
- Have Wax build pages from the JSON instead of the .csv

Title

Minard's Map of Napoleon's March

Creator

Charles Minard

Date of Creation

1869

Source URL

upload.wikimedia.org

Description

Charles Minard's visualization depicting Napoleon's march on Moscow

Example Type

data visualization

Visualization Types

- line chart
- map / spatial data

Subject Area

war



Submit a teaching example! (Responses) ☆ 📄 ☰

File Edit View Insert Format Data Tools Form Add-ons Help Teach Vis By Example La

100% \$ % .0 .00 123 Default (Ari... 10 B I S A

	A	B	C	D	E
1	pid	approved_date	timestamp	label	creator
2	dv1	6/26/20	5/12/2020 14:02:11	Minard's Map of Napoleo	Charles Minard
3	ds1	7/10/20	5/12/2020 14:07:27	Time to Statham punch	Matt Haughey
4			5/12/2020 14:14:57		DPerezdelangel09
5	ds2	7/10/20	5/12/2020 14:33:30	United States of America Van Panhuis W., Cros	
6	ds3	7/10/20	5/12/2020 14:46:37	Estimated Use of Water i	Dieter, C.A., Linsey, K
7	ds4	7/10/20	5/12/2020 14:46:56	Seattle Pet Licenses	City of Seattle: FAS -
8	dv2	6/26/20	5/12/2020 14:47:34	Gun Deaths in Florida	C.Chan
9	dv3	6/26/20	5/12/2020 14:47:43	built : Los Angeles	Andy Rutkowski
10	ds5	7/10/20	5/12/2020 14:47:54	Good, Evil, Ugly, Beautif	Gregor Aisch, Damon
11					
12	dv4	6/26/20	5/12/2020 14:49:51	Casualties of the Iraq Wa	Simon Scarr
13	dv5	6/26/20	5/12/2020 15:01:50	Water Use in the U.S.	USGS VIZLAB
14	ds6	7/10/20	5/12/2020 15:24:32	Averaged Gapminder dat	Gapminder
15	dv6	6/26/20	6/8/2020 13:10:57	Legislative Explorer	John Wilkerson and F
16					
17					
18					
19					
20					
21					
22					
23					

+ ☰ Form Responses 1 10 Manually Cleaned Responses datavis data

Clean submissions

File Edit View Run Publish Resources Help

Select function

Code.gs

```

1 function onOpen() {
2   var spreadsheet = SpreadsheetApp.getActive();
3   var menuItems = [
4     {name: 'Clean List Columns', functionName: 'cleanListColumns'}
5   ];
6   spreadsheet.addMenu('Teach Vis By Example', menuItems);
7
8   cleanListColumns();
9   //Note: can't run the refresh automatically on load;
10  //have to run from menu or from script editor because of URL fetch
11  //parseRAMLtriggered();
12 }
13
14 function cleanListColumns() {
15
16   var cols_vals = {
17     audience_level: ["Young beginner (pre-high school)", "Adult beginner (high sc
18     audience_composition: ["General", "Librarian/information professional"],
19     language_tool: ["R (e.g., ggplot2, leaflet)", "Python (e.g., seaborn, bokeh)"
20                   "Google Sheets / Slides / Docs", "Tableau", "JavaScript (e.g.
21     data_type: ["1 or more categorical variables", "1 or more numerical variables
22     vis_type: ["bar chart", "line chart", "pie chart or donut chart", "scatter plot
23   }
24
25   var spreadsheet = SpreadsheetApp.getActive();
26   var sheet = spreadsheet.getSheetByName('Manually Cleaned Responses');
27   sheet.activate();
28
29   var submissions = sheet.getRange(2,1,sheet.getLastRow()-1,23).getValues();
30
31   // audience_level is column 11
32   // audience_composition is column 12
33   // language_tool - 16

```

How we are extending Wax: new template for multi-facet filters

- Our use case is really focused on information retrieval, rather than gallery browsing
- Default filter functionality in Wax collection gallery operates for single field, single selection at a time
- Have been working on new checkbox-style filters that allow for selections of multiple values from multiple fields



Filters

Subject Area

- ☒ geography
- ☒ housing
- ☒ hydrology
- ☒ law
- ☒ safety
- ☒ war

clear all

select all

Visualization Types

- ☒ animation
- ☒ bar chart
- ☒ dashboard
- ☒ infographic
- ☒ line chart
- ☒ map / spatial data
- ☒ pie chart or donut chart

clear all

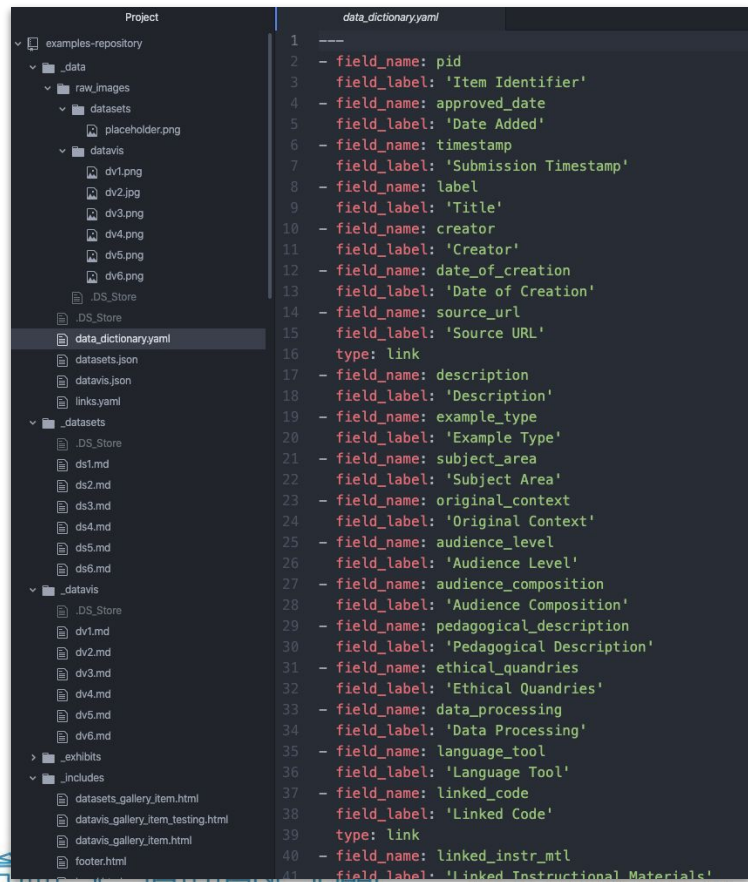
select all

How we are extending Wax: Data dictionary

- For some of our website functionality, we needed a way to match up item metadata fields with helpful information about those fields
- Created a data dictionary in YAML format that gets associated with collections and used in different layouts and includes
- Current meta-metadata:

○ field type (list, URL)

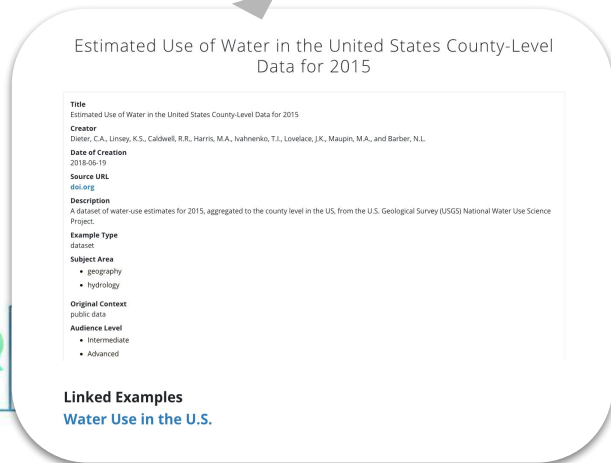
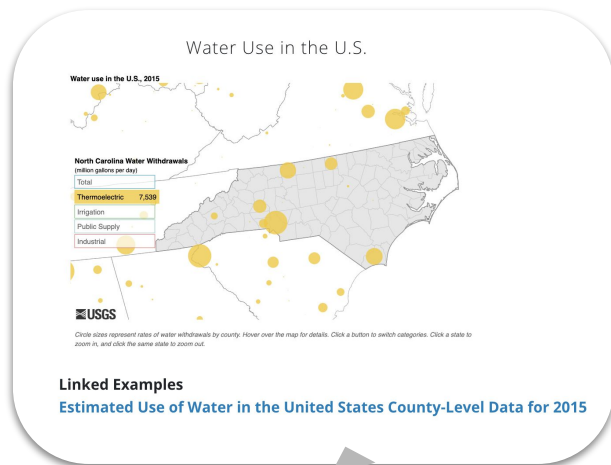
○ human-readable name



```
1 ---
2 - field_name: pid
3   field_label: 'Item Identifier'
4 - field_name: approved_date
5   field_label: 'Date Added'
6 - field_name: timestamp
7   field_label: 'Submission Timestamp'
8 - field_name: label
9   field_label: 'Title'
10 - field_name: creator
11   field_label: 'Creator'
12 - field_name: date_of_creation
13   field_label: 'Date of Creation'
14 - field_name: source_url
15   field_label: 'Source URL'
16   type: link
17 - field_name: description
18   field_label: 'Description'
19 - field_name: example_type
20   field_label: 'Example Type'
21 - field_name: subject_area
22   field_label: 'Subject Area'
23 - field_name: original_context
24   field_label: 'Original Context'
25 - field_name: audience_level
26   field_label: 'Audience Level'
27 - field_name: audience_composition
28   field_label: 'Audience Composition'
29 - field_name: pedagogical_description
30   field_label: 'Pedagogical Description'
31 - field_name: ethical_quandries
32   field_label: 'Ethical Quandries'
33 - field_name: data_processing
34   field_label: 'Data Processing'
35 - field_name: language_tool
36   field_label: 'Language Tool'
37 - field_name: linked_code
38   field_label: 'Linked Code'
39   type: link
40 - field_name: linked_instr_mtl
41   field_label: 'Linked Instructional Materials'
```

How we are extending Wax: Connections between examples

- Using the data dictionary, we can specify a metadata field as a special type: “internal”
- Internal fields contain lists of identifiers to other examples in the repository
- On item pages, those internal identifiers get converted to pretty links to the related examples



How we are extending Wax: support for list fields

- Create metadata in spreadsheet (.csv)
- Use python script to split lists into arrays and output JSON
- Have Wax build pages from the JSON instead of the .csv

Title

Minard's Map of Napoleon's March

Creator

Charles Minard

Date of Creation

1869

Source URL

upload.wikimedia.org

Description

Charles Minard's visualization depicting Napoleon's march on Moscow

Example Type

data visualization

Visualization Types

- line chart
- map / spatial data

Subject Area

war



How we are extending Wax: new template for multi-facet filters

- Default Wax filter: one field, one selection
- Supporting multiple fields, multiple options:
 - support list fields
 - create new page template with new interface elements, filter logic
 - polish user experience (ongoing)

Filters

Subject Area

- ☒ geography
- ☒ housing
- ☒ hydrology
- ☒ law
- ☒ safety
- ☒ war

clear allselect all

Visualization Types

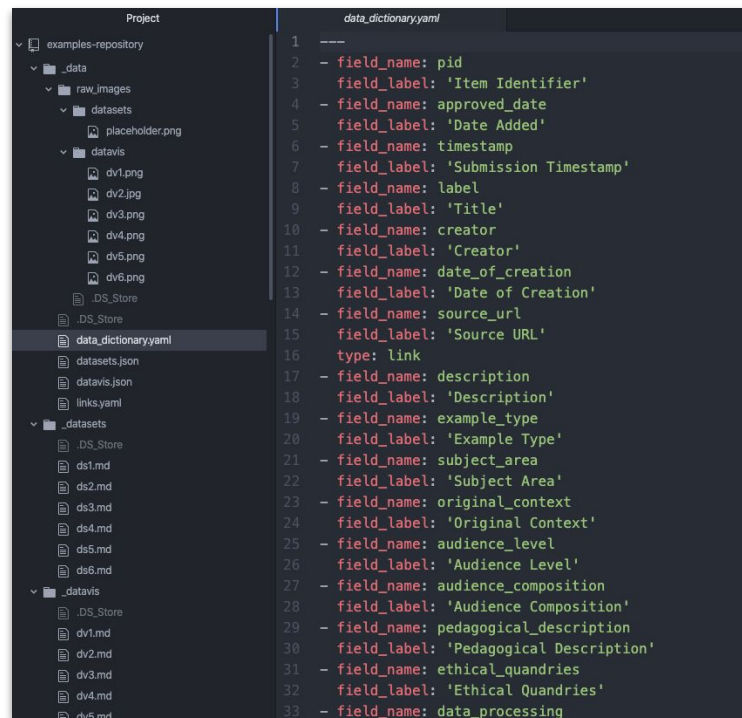
- ☒ animation
- ☒ bar chart
- ☒ dashboard
- ☒ infographic
- ☒ line chart
- ☒ map / spatial data
- ☒ pie chart or donut chart

clear allselect all



How we are extending Wax: Data dictionary

- Create data dictionary in YAML format
- Use Jekyll config file to connect collection to data dictionary file
- Current meta-metadata:
 - human-readable name
 - field type (list, URL)



How we are extending Wax:

Connections between examples

- Create special field type: “internal”
- Create lists of identifiers of connected examples
- Design item pages to convert IDs to pretty links

